LiDAR Augmented Optimal Path Planning for Unmanned Aerial Vehicle during Remote Sensing at Low Altitude and Network Sensor Data Acquisition

Technical Report

ARCAA-RS-2012-01

Australian Research Centre for Aerospace Automation

Y.Y.M. Leong

Queensland University of Technology, Brisbane, Australia

***Abstract* – This technical report describes a Light Detection and Ranging (LiDAR) augmented optimal path planning at low level flight methodology for remote sensing and sampling Unmanned Aerial Vehicles (UAV). The UAV is used to perform remote air sampling and data acquisition from a network of sensors on the ground. The data that contains information on the terrain is in the form of a 3D point clouds maps is processed by the algorithms to find an optimal path. The results show that the method and algorithm are able to use the LiDAR data to avoid obstacles when planning a path from a start to a target point. The report compares the performance of the method as the resolution of the LIDAR map is increased and when a Digital Elevation Model (DEM) is included. From a practical point of view, the optimal path plan is loaded and works seemingly with the UAV ground station and also shows the UAV ground station software augmented with more accurate LIDAR data.**

Keywords – UAV, Optimal Path Planning, low level flight, LiDAR

I. INTRODUCTION

The use of UAVs for remote sensing civilian applications such as air sampling or early detection of bushfires is becoming increasingly popular. One of the problems that arise when performing tasks such as air sampling or data acquisition from ground-based sensors is the need to fly at low altitudes to collect data. The UAV has to be able to fly the shortest path between certain points, while avoiding obstacles that are present at low altitudes. One challenging remote sensing task is for a UAV to monitor inaccessible cultivation areas and sample air and look for either unwanted spores or other plant pathogens. An UAV fitted with such data collection system and air sampling device flying an optimal path can monitor and reduce the risk of pest introduction from international trade and, at the same time, will capture a wide range of plant health information in a cost-effective way so as to cover international and domestic market demands. It can also collect data form temperature sensors, georefenced images and Multispectral images to monitor possible bush fires and reduce the risk of vegetation encroachment under power lines [1, 2].

In this report , we consider the UAV with the given task of flying through a set of fix set of nodes from a start point to an end point on the map at low altitude. The UAV has to fly on the shortest path to reach the target, while avoiding all obstacles. Two missions are considered; one is when the UAV has an air sampling device and is commanded to sample at different waypoints. Here, the data can be sent back to the ground station in real time. The second mission is one in which the UAV has to find an optimal path, fly within a certain range of several predefined ground sensor nodes, and fly through the nodes in a certain order. When within range, each sensor node will transmit its data to the UAV. Once data has been received from all the nodes, it can be either transmitted in real-time to the ground station or processed when the UAV lands at the base station. This concept is illustrated in Figure 1.

The sensor nodes are gas sensors which can be used for a variety of functions. For example, it can be used by farmers to monitor gas emissions from crops or used for early detection of forest fires.

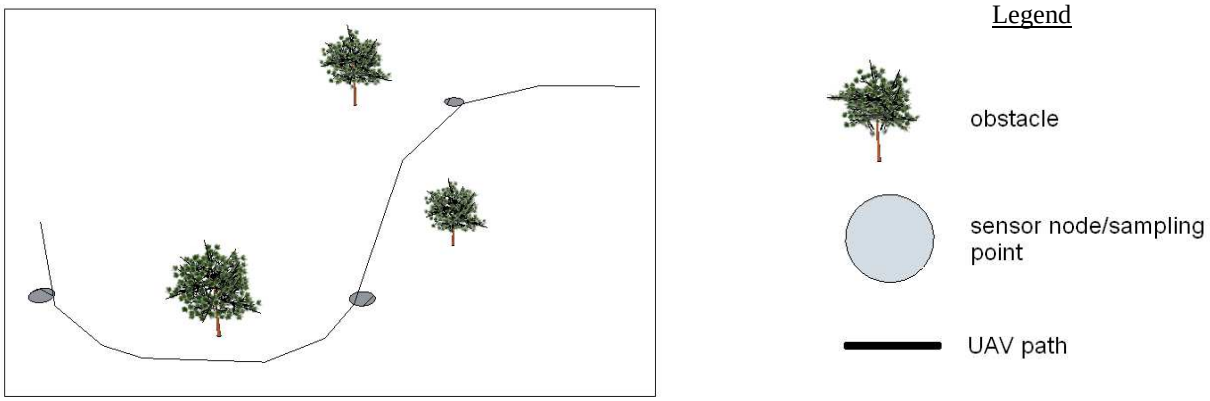

Figure 1 Illustration of concept; low level flight air sampling mission

The use of UAVs for remote sensing and for vegetation monitoring using thermal and narrow-band multispectral concepts has been studied by a number of researchers [3, 4]. In this work we augment the knowledge of the mission area by adding a digital map to the ground station. One important aspect of a UAV is the design of an optimal path plan and trajectory. Traditionally optimal path plans are found using deterministic optimisers but this may be trapped in local minima [5,6,7,8,9,10]. Other techniques such as evolutionary algorithms are robust to find global solutions but suffer from large computational expense;

therefore, one of the main objectives in optimal path planning is to develop effective and efficient optimization techniques in terms of computational cost and solution quality [11].

One of the problems that the UAV may have is obstacles that it has to avoid along the mission. The application of path planning and obstacle avoidance for a UAV or ground robot has been explored by a number of researchers [12]. There are a number of complex algorithms that may be used for UAV path planning [13]. In this work, a modified implementation of the A-Star (A*) algorithm [14] is used for illustrative purposes but more complex algorithms are being explored [5-10,15] . A UAV is usually controlled by a ground station where an image of the area in which the UAV flies is known and is loaded to the ground station computer. Due to the number of possible low level obstacles on an air sampling and data collection UAV low level terrain data is therefore needed.

The rest of the report is organized as follows; section 2 describes the form of terrain data used in this research, section 3 describes the method and algorithm, section 4 describes the implementation, section 5 discusses practical test cases, conclusions are presented in section 6.

II. LIGHT DETECTION AND RANGING (LIDAR) AND DSM GENERATION.

A digital elevation model (DEM) is a digital representation consisting of terrain elevations for ground positions at regularly spaced horizontal intervals or grids. DEM may be used in the generation of three dimensional maps displaying terrain slopes and terrain profiles in diverse regions. Three dimensional maps can offer a representation of both ground surface and the objects of that plane. Such representations are commonly named as Digital Surface Models (DSM), having information about buildings, roads, vegetation and terrain features [16]. DSMs are commonly used to create 3D Terrain Flythrough scenes, augment simulated environments, and for diverse spatial applications. DEM and DSM can be obtained for diverse sources such as remote sensing techniques, surveillance measurements, photogrammetry, topographic maps, and so on. Indeed DSM are freely distributed by Terra (EOS AM-1) satellite provided in ASTER format and other distributors such as the data from The Shuttle Radar Topography Mission (SRTM), both with a grid resolution limited from 30 meters to 90 meters according with the region[17,18, 19] Such footprints clearly can not represent some of the obstacles with enough accuracy to perform close range path planning; to overcome this problem this experiment have used data from an Airborne Light Detection and Ranging Laser Scanner (LiDAR).

LiDAR is a remote sensing technology that uses laser light reflections to determine the position or other characteristics of a distant target. Here, LiDAR technology was used to perform airborne scanning for three dimensional (3D) mapping, which will create a point cloud of the earth's surface. Each point contains a set of parameters pertaining to that single point on the ground. Typically, these parameters will be the x-coordinates, y-coordinates, elevation, number of returns, intensity, and will be saved in the LAS file format [19, 20]. For this application which requires remote sensing data of forested areas, the usage of LiDAR data is beneficial as it can record multiple returns with each outgoing pulse. Furthermore, the data was provided classified in ground and non-ground, being possible to differentiate objects (buildings, vegetation, etc) and bare terrain (see

Table 1 for more details).

TABLE 1. DESCRIPTION OF THE LIDAR DATASETS USED

Place	Point Density	Classifications	Min elevation (AMSL)	Max Elevation (AMSL)	Height	Width	Used in Test case #	Instrument
Site 1	4 pts/m^2	Ground/Non Ground	412 m	461 m	447 m	142 m	1-5	RIEGL LMS-Q560
Site 2	10 pts/m^2	Ground/Non Ground	165 m	195 m	1000 m	459 m	6-8	Leica ALS50

In contrast with the satellite DEMs, an ALS LiDAR can provide data with up to less than 10 cm of resolution, depending of the height of the flight and specifications of the instrument. This implies to have millions of points (and hundreds of MB) for each single data tile; consequently a high computing cost to process this data is needed. In fact, there are several techniques to reduce the resolution of the LiDAR without losing fundamental terrains characteristics or objects [20,21] , this helps substantially to reduce the processing time and increase the efficiency of the computations. In this experiment a thinning algorithm has been applied using a toolbox created for Martin Isenberg called “LAStools”. After the resolution of the 3D point cloud data was reduced it was required to generate a DSM in USGS format in order to upload it to the UAV simulation software). To overcome this, we used GRASS and 3DEM software, both open source distributions. Figure 2 shows a top view of one of the sample LIDAR maps used in this research, with points coloured by elevation. The image was selected as it has a large vegetation area with a good number of obstacles. Figure 3 is the three dimensional (3D) view of the same terrain. The terrain form the LAS format can be viewed in 3D with the appropriate software (GRASS, lastools, MARS viewer). This LAS file is then converted to a text file which lists each point’s parameter values. This text was then processed to be used by the path planning algorithm. shows an example of a portion of the text file.

Figure 2. DSM used for Test cases 1-5

Figure 3 DSM used for Tests cases 6 and 7

The LAS file was then converted into a text file using 'LAStools'. With this program, the parameters that are printed in the text file and the order in which they are printed can be selected as required. In this manner the data can be used by the path planning algorithm. In Figure 5 below, the parameters from left to right are *x*, *y*, *z*, intensity and classification

```
293929.747 6125833.882 433.047 91 5
293929.779 6125830.895 451.688 42 5
293929.826 6125831.066 450.135 29 5
293929.699 6125833.542 428.697 409 2
293927.79 6125833.507 431.621 39 5
293929.304 6125833.609 428.657 336 2
```

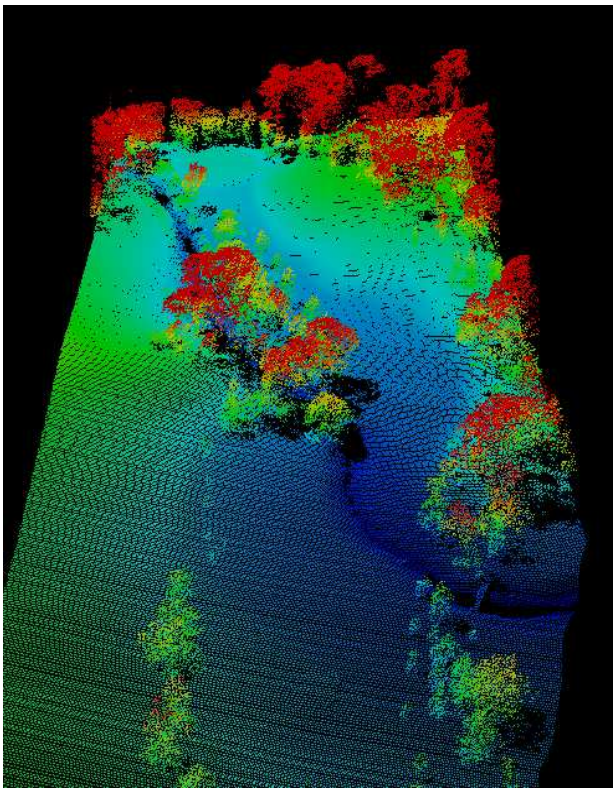

Figure 5. A top view of the site 1 sample LIDAR maps used in this research, with points coloured by elevation.

III. PATH PLANNING ALGORITHM

There are several algorithms that can be used for the purpose of UAV path planning [5-10, 15]. Here, the use of the A-Star (A*) algorithm for UAV path planning is considered for illustration purposes, but more complex algorithms can be used. The A* algorithm is a popular choice for path finding as it is quite flexible and can be used in a variety of contexts. In path planning, the area to be searched is simplified by dividing into several smaller areas, which can be of any shape, such as rectangles or triangles. However, for simplicity, a square grid is used. This square tile is also known as a vertex. The A* algorithm is a combination of the two principles of Djikstra's algorithm and the Best-First-Search (BFS) algorithm [15].

Djikstra's algorithm works by continuously examining vertices closest to the starting point and expands outwards until it reaches the goal. The algorithm will then find the shortest path from the starting point to the goal. The BFS algorithm, on the other hand, utilizes an estimate (called a heuristic) of the distance between the current vertex from the goal, and uses this to select vertices closer to the goal, instead of selecting a vertex closest to the starting point. This enables the algorithm to search through fewer vertices. However, the path found will not necessarily be the shortest. In contrast, Djikstra's algorithm has to search through more vertices, but will guarantee the path found is the shortest. Thus, the A* algorithm is a combination of the two desirable properties of these algorithms.

For each vertex or node traversed, three values are stored. They are related by equation (1).

$$f(x) = g(x) + h(x) \qquad \text{Eq. 1}$$

Where *g(x)* is the cost of the path from the starting point to the current vertex, h(x) is the estimated distance from the current vertex to the goal (or heuristic) and *f(x)* is the sum of both.

To begin, the starting square A is added to a list of squares to be examined, called an 'open list'. All the reachable squares adjacent to square A are also added to the list. For each of these squares, square A is the 'parent' square, and each square stores a pointer pointing back to its parent square. Then, one of these adjacent squares is chosen, the square is switched to the 'closed list' (list of squares that do not need to be examined), and the process is repeated again.

To decide on which square to choose, equation 1 above is used, and the square with the lowest cost is chosen. The cost is one for moving in a horizontal and vertical direction, and $\sqrt{2}$ for moving diagonally. In this example, for simplicity these values are multiplied by 10 and replaced with 10 and 14 respectively. These costs can be adjusted as necessary, depending on the application. The estimated cost of the distance from the current point to the target is calculated here using the Manhattan method. This method calculates the total number of squares moved horizontally and vertically, ignoring diagonal movement and any obstacles. As above, the cost is then also multiplied by 10. Once the next square is chosen, the process is repeated on the next adjacent squares, ignoring any obstacles and squares already in the closed list. This process is repeated until the target square is added to the closed list (target has been found). Then, from the target, the pointers are followed to obtain the path travelled from the starting square. If the target is not found and the open list is already empty, then there is no path to target square.

IV. NUMERICAL IMPLEMENTATION

In this work a modified C++ implementation of the A* search algorithm described in [15] was used. Several LAS files were also obtained for testing purposes. The task was to design and implement a C++ program to receive data from the LAS file and process it to become a suitable format before sending it to the A* algorithm program. The A* program will then calculate and generate the coordinates of the shortest path to be taken. Once the path is obtained, it was saved back into the LIDAR map, to be viewed together with the terrain. Such terrain with the path also is then processed to be able to be viewed in MATLAB software. This program will also convert the optimal path plan into a set of waypoints suitable for typical UAV autopilot and ground station software. The implementation process is best described as a flowchart depicted in Figure 6.

The first step is to convert the LIDAR map from a LAS file format to a text file. The second step is to process the points in the text file so that the A* algorithm can be applied. Next, the start, end and node positions are defined (step 3) and the optimal path is obtained using the A* algorithm (step 4). Step 5a and 5b are used to visualize the processed terrain data in both the LAS file format and in MATLAB. Step 6a and 6b involve plotting the obtained optimal path in both the LIDAR map and MATLAB terrain. In Step 7, the LIDAR map is loaded as an image or DEM (See Section II) in the UAV ground station software. Step 8 will load the optimal waypoints to the autopilot ground station and simulate the mission.

V. PRACTICAL TEST CASES

Seven test cases of increasing complexity were considered in this work. The first test case uses a reduced map size, rounded up integer coordinate values and has start and end points only. The second test case uses floating point coordinate values only having start and end points. Test case three involves plotting the path through multiple fixed nodes (representing the ground sensor or air

sampling node points) between start and target points. The fourth test case calculates and plots an optimal path in an enlarged map. Test cases 5 and 6 involve the UAV ground station and waypoint navigation. Test case 5 is the flight simulation on the enlarged map from the test case 4. Test case 6 uses a second larger map for flight simulation and shows the UAV sampling at predefine locations. Test 7 details an air sampling/data collection at predefined locations.

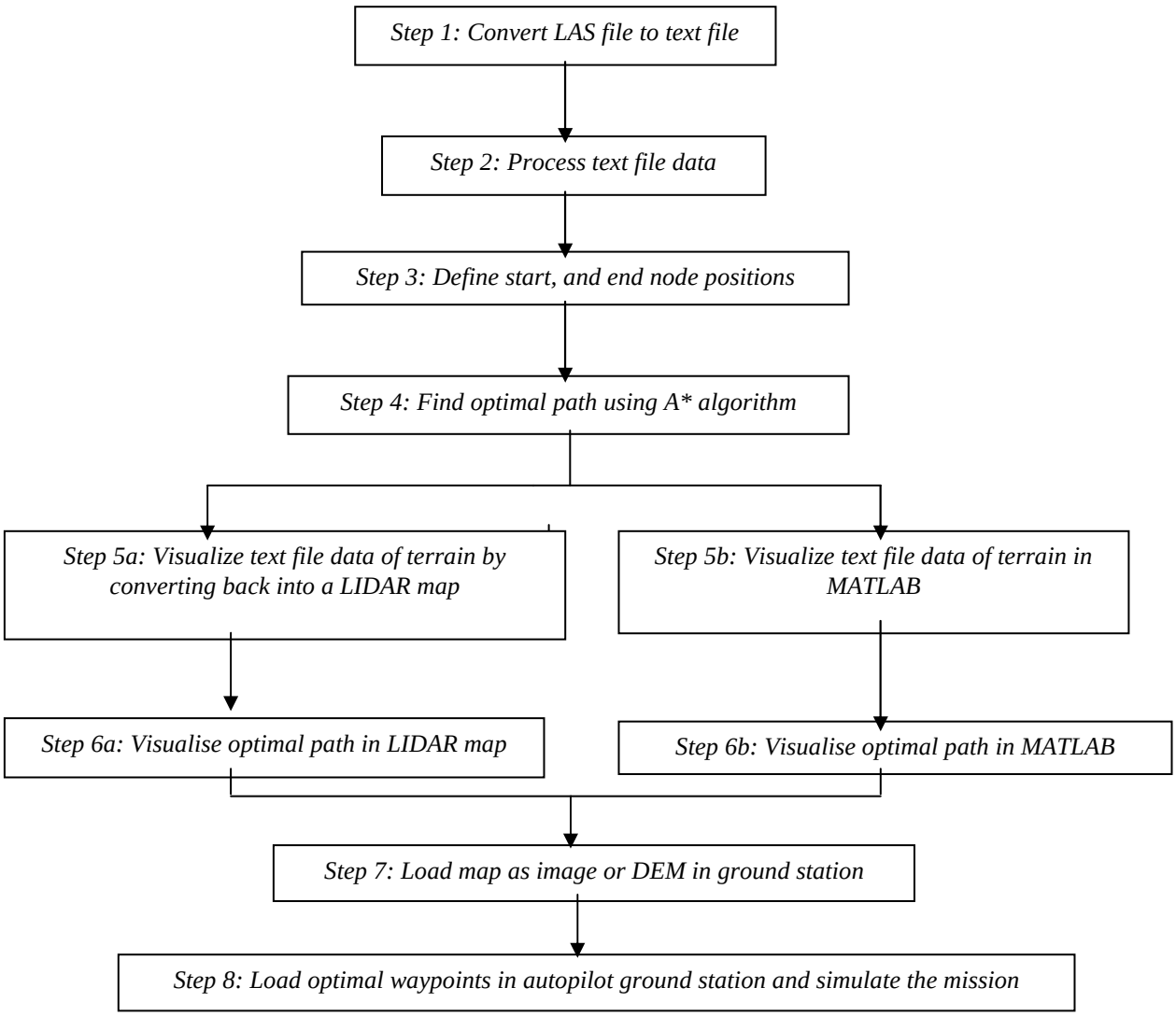

Figure 6. Flowchart of implementation process

A. Test Case 1: Reduced map size, rounded up coordinate values, start and end points only

In this case the aim was to first experiment with the algorithm and a LIDAR file to determine if the concept and procedure described in Figure 6 was feasible. Since the path planning algorithm is a two-dimensional (2D) path planner, it was necessary to make an adaptation in order to apply the A* 2D methodology to the 3D point clouds from LiDAR data; Assuming that the UAV is meant to fly at constant height (close to the ground level for surveillance proposes) during all the mission, a set elevation of 427 meters AMSL was selected since it represents few meters above the maximum terrain elevation. To adapt the 3D point clouds to the 2D scenario, all the points below 427 meters were cropped from the map. Then a 2D grid with elevation of 427

meters was formed, a cell was occupied for an obstacle if one or more of the remaining 3D points were within the range of those cells (easting and northing position). Furthermore, all point clouds below the fly planning were eliminated and translated to a 2D grid. In this way, the C++ source code was written to process the data points before sending it to the A* algorithm program. Consequently a 2D array was created based on the 2D grid explained before; this array stored a '1' if that point was an obstacle, and a '0' if it was free space. If the cell is occupied it is considered an obstacle. Each coordinate were indexed to start from zero, For example, the point with the lowest *x* and *y* coordinates will be represented in the location of [0 0] in the array and so on. The A* program requires the start and end nodes as well as the LIDAR input in the form of a text file which has an array of ones and zeros, with ones indicating there is an obstacle. Thus, the 2D array of ones and zeros that has been formed in the step above has to then be saved in another text file. Path planning is performed (step 4) and the result is shown in an output file (Figure 6 illustrates the process) .

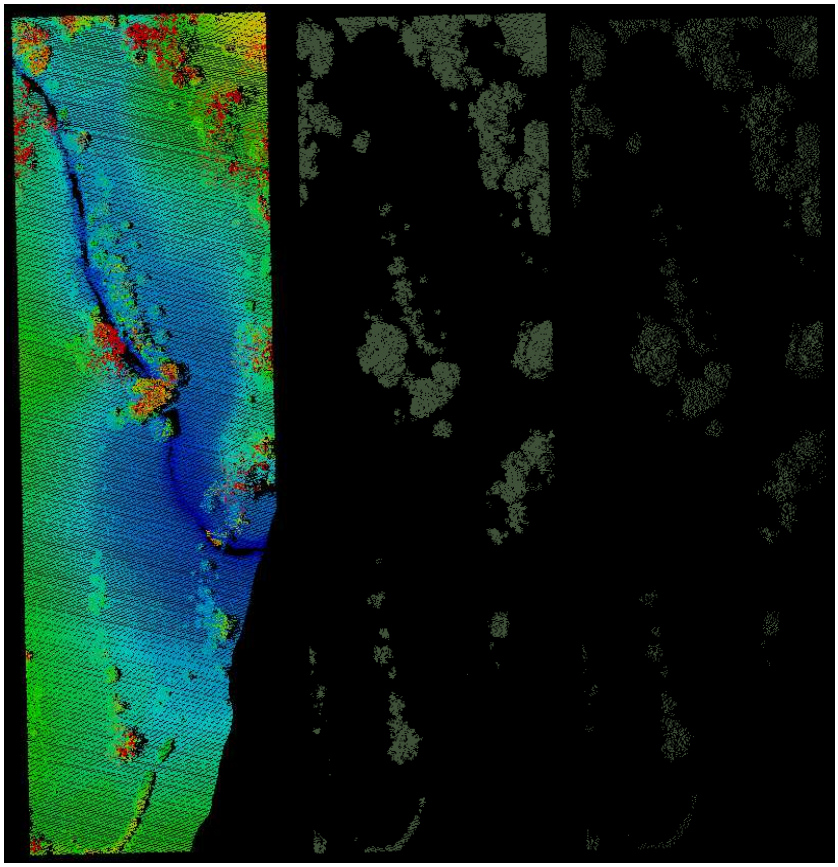

Figure 7. Original map, exact coordinate values, start and end points only

The path travelled is given as coordinates in the output file, which can then be plotted in MATLAB. The resulting figure is then overlaid on top of the LIDAR map to observe the plotted path. In this first test case, steps 5 onwards were not implemented.

B. Test Case 2: Original map, exact coordinate values, start and end points only

The first test case illustrated that the method is feasible; the algorithm was therefore implemented using exact coordinates (with decimal points). The LAS data file is used and the data is converted into a text file. Here, only points with a *z* value of more than 427m were used. One of the problems that arises when trying to use the exact coordinates and resolution of the original LAS file,

is that the range of values is too large; there are too many points of data and not enough memory (Intel Core 2 6300 , 1.86 GHz x2CPUs) to store all values in an array. To solve this problem, initially one in every five lines in the text file was extracted. The resulting text file was then converted back into a LAS file to observe its effect. Figure 16 a, b and c compare the original map (Figure 16a) with a map that only contains z coordinates larger than 427 m and one that has been filtered, with only one in five lines from the original LiDAR LAS file being used. However, it was decided that this will not solve the problem, as even though there are less points that represent an obstacle, the dimensions of the map are still the same. Thus, the range of values is still very large and a large array will still be required to represent the map. One other alternative is to use the classification information of the LAS files in order to discriminate ground and non ground an reduce the amount of point to be analyzed. The points in the text file indicate where there are obstacles only, but the array has to represent both obstacles and free space, so that the algorithm knows where it can plot the path.

The next method used to attempt to solve the problem of PC memory required was to decrease the resolution, by grouping the values. This is done by representing every '*x*' number of points as one value in the array. Initially, every 100 points from the text file were stored as one value in the array. However, as the results were not satisfactory, it was changed to every 1000 points instead. The effect of decreasing the resolution on the map can be seen in the result in the figure . The size of the file is reduced and fewer points are needed to represent the map.

Results with 1000 points stored as one value in the array proved to be satisfactory. The procedure described in figure 11 is followed; an optimal path is found and then inserted back into the 2D LIDAR map. This is done by saving the coordinates of the path in the text file before converting it back in LAS format. The path's points are given a different classification so that it can be differentiated from the terrain when viewed. The path can also be inserted back into the original 3D map (with the path being at constant height, *z* of 427). The 3D terrain and path is also visualised in MATLAB as shown in figure 8.

Table 2 shows the computation times of the various tasks performed during the implementation. Steps 6, 7 and 8 were not considered in this test case.

Task	**Computational time (s)**	**% of time**	**Processor**
Step 1 – Convert LAS file to text file	2.015	18	Intel Core 2 6300 (1.86 GHz x2CPUs)
Step 2 – Process text file data	1.465	13	
Step 4 – Find optimal path using A* algorithm	0.036	0.3	
Step 5 – Convert values from array format to text file format and MATLAB coordinates format	2.283	20	
Step 5 – Convert text file to LAS file	1.218	11	
Step 5 – Convert and display terrain with path in MATLAB	4.282	38	
Total	11.299		

Table 2 Computation times for steps 1-5

It can be seen that a large percentage of the total time spent is used for displaying the path in MATLAB (38% of total time). When this program is run on a UAV ground station or onboard computer the path display is not needed, the computation time will be reduced substantially. Conversion of the LIDAR file to a text file format takes up 18% of the computation time, conversion for visualisation in MATLAB takes 38 % of the computational time, while the actual task of finding the optimal path using the A* algorithm only takes 0.3% of the total computation time.

A. Test Case 3: Path planning through multiple fixed nodes between the start and target points

The next step was to add nodes between the start and target points representing the predefined ground sensor or air sampling node points. The program calculates an optimal path from the start point which passes through each node and ends at the target point. This is done by applying the A* algorithm between the first two points to obtain a path, then repeating this for each pair of points until the target is reached. With this method, the path between each pair of points is found independently without taking into account the other points.

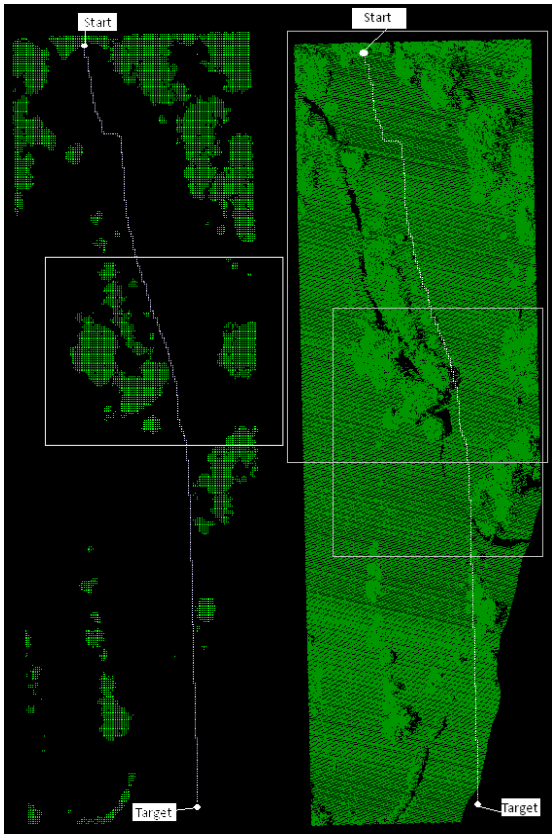

Figure 8. 2D LAS File Result

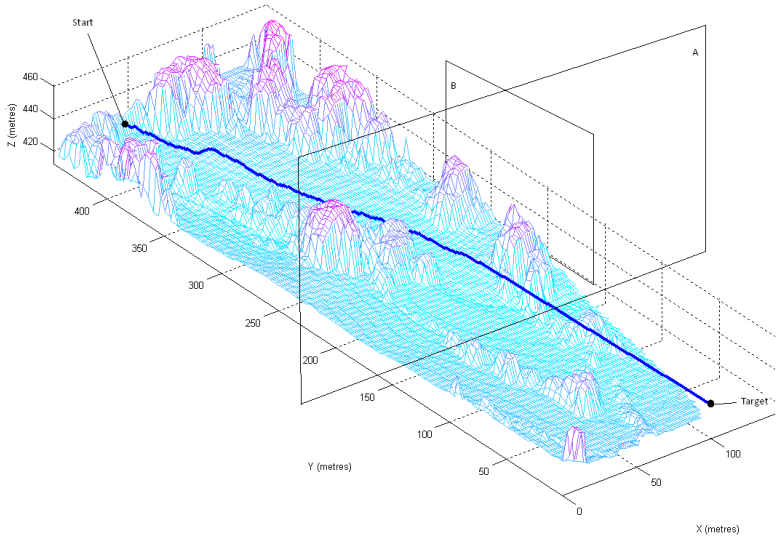

Figure 9 MATLAB 3D results

The input file to the A* algorithm program contains the map itself represented as an array of ones and zeroes, as well as the size of this array, the number of nodes and the coordinates of the start, target and node points (converted from map coordinates to row and column values). The number of node coordinates depends on the number of nodes required. Figure 9 shows the resulting path found when three nodes are placed in between the start and target points. The program first finds the shortest path between the start point and node 1, then from node 1 to node 2 and so on until it reaches the target.

A. Test Case 4: Path planning in enlarged map

From a practical point of view, it was found that when a small LIDAR map (400 x 400 m) is used during simulation of a UAV's flight, the map was too small for the UAV to be able to navigate due to its dynamic characteristics and small turning radius To solve this issue, it was decided that a larger LIDAR map and a helicopter UAV would be used for the mission. For testing purposes, it was decided that the size of the original map would be increased instead by multiplying the same map four times and mirroring the bottom portion.

Another practical problem is to take into account the dimensions of the plane. For this reason, the grid size used for the implementation of the A* algorithm needed to be increased. This is done by decreasing the resolution by reducing the number of points used to represent the map. Then, instead of having a point every meter, there will be one point every five meters instead, while still keeping the LIDAR map dimensions constant. This gives the aircraft an area of five square meters representing the UAV rotor or wing span plus a margin of safety. The grid size on the map is changed by changing the 'factor' variable. This variable determines the amount of rows and columns to be used when converting from LiDAR coordinates to the array form of row and columns. When the factor is increased, less rows and columns will be used to represent the same map, and thus the resolution is decreased. Each LIDAR map point is separated evenly five meters apart, this gives a safety marging for the UAV to fly trough.

A. Test Case 5: UAV ground station and waypoint navigation (using enlarged map from Section 4.4)

One important aspect of the method is to determine if the optimal path and system is feasible and compatible with the UAV hardware and software. Flight simulations were performed using the UAV Ground Control Software [24]. In this application, the UAV has to fly and obtain data at low altitudes while avoiding obstacles. As discussed in the previous sub-section, preliminary simulations indicated that the turning radius of one of our fixed wing UAV is not small enough to meet these demands. Thus, it was decided that a helicopter UAV would be more suitable

The helicopter UAV simulation model used is based on the Xcell60 helicopter. Some of its specifications are listed in the table below [23]

Manufacturer	Miniature Aircraft USA
Main Rotor Diameter	57.25 in (1455 mm)
Length	53.5 in (1360 mm)
Height	16.25 in (413 mm)
Tail Rotor Diameter	11.1 in (282 mm)
Main gear ratio	9.0:1
Main to tail gear ratio	1:4.37
Weight	9lb 12oz. (4.43 Kg)
Power plant	0.61 cu. in. (10 cc) two-stroke glow plug engine
Control Requirements	5 servos and a gyro

Table 3Xcell 60 specifications

First, the source code was amended to make the program automatically generate an output file which consists of a set of coordinates from the optimum path found as well as commands for simulating the aircraft (Figure 6 Step 6) . These coordinates are converted from the Universal Transverse Mercator (UTM) form of Northing and Easting values to a Cartesian coordinate form required by the ground station. They consist of the start, target and node coordinates, as well as a user-defined number of coordinates between each node point. The flight was then simulated and the results are shown in the figures 10 . In this first test case, the coordinates obtained when using the enlarged map are used.

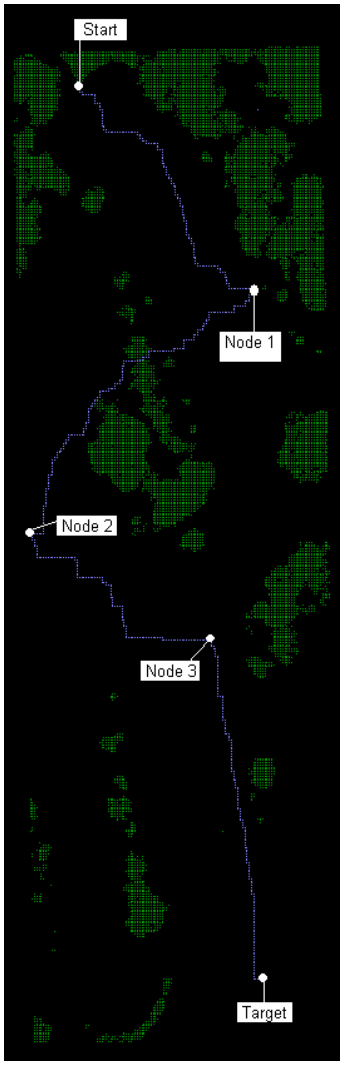

Figure 10 Result of path with predefined ground sensor or air sampling node point

The UAV takes off at its origin, flies to the first waypoint, and then continues on to the next waypoints towards the last waypoint at the bottom right. Once it reaches the last waypoint, it proceeds to land at the origin. It can be seen that there is some oscillation in the flight path at the first four waypoints. This is due to the fact that in this case the helicopter's origin was at the bottom left of the map, resulting in the helicopter having to make an 180˚ turn near the first waypoint in order to start following the path. The rest of the flight is consistent, with only a slight margin of error.

B. Test Case 6: Flight simulation on exact location

A larger LiDAR map, with its exact location known, was then obtained, an optimal path was found and the flight simulation was performed and Figure 11 show the results of this simulation.

It can be seen that the helicopter is able to follow the optimal path successfully up until the set of waypoints, due to the waypoints being too close together. This is related to the dynamic capabilities of the aircraft and navigational capabilities of Horizon[mp] (the ground station software).

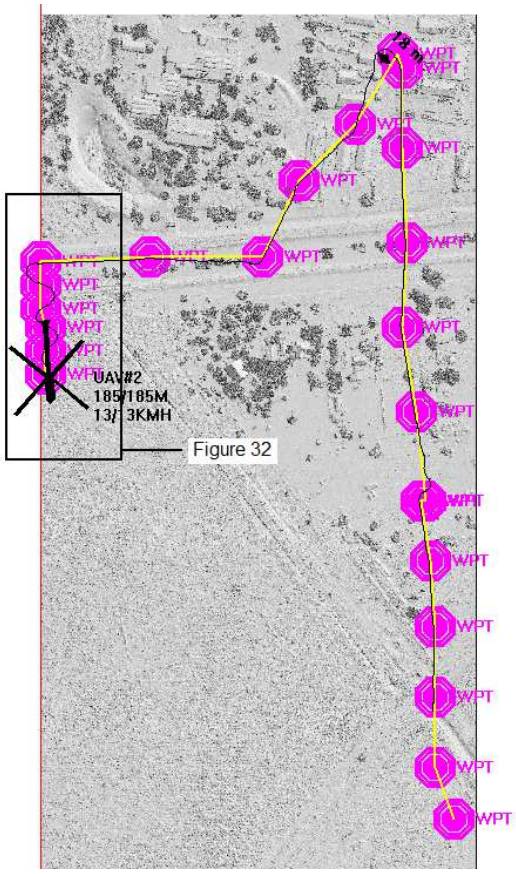

Figure 11. UAV path.

Specific to the software to solve this problem is the testing of different navigation commands. In previous tests a fly-*to* waypoint command was used, which directs the helicopter to fly from one waypoint to another continuously. A *hover-at* command is used instead. This command instructs the helicopter to fly to one waypoint and then hover at that point for a predetermined period of time while adjusting its position to point towards the direction of the next waypoint. This rectifies the problem of the oscillating flight path. Simulation results are shown in Figure 12.

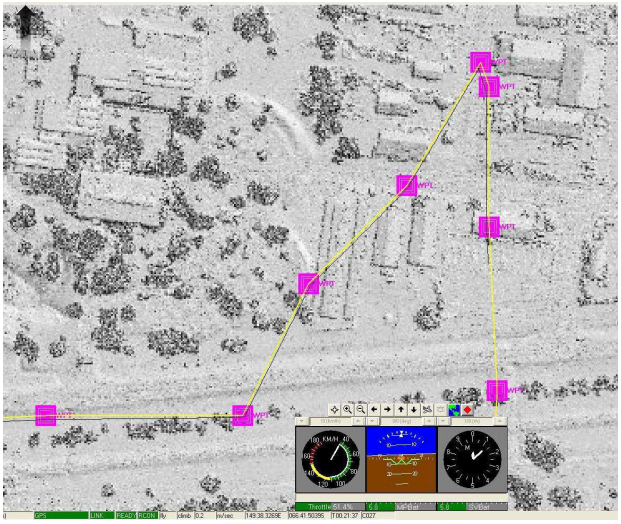

Figure 12. Optimal flight path with –hover-at command

C. Test Case 7: Air sampling/data collection at predefined location

This example shows the complete system and procedure defined in section IV and figure 11. LiDAR data is processed, an optimal path is found and sent to the autopilot for the UAV to sample and collect the data.

In this test we simulate the flight while considering the application of data acquisition and air sampling. The hover-at command is used at predetermined waypoints where it is required that the helicopter UAV receive data. This enables the helicopter to hover at that waypoint for the amount of time required to successfully receive the data. This time will also serve the added benefit of allowing the helicopter to rotate towards the next waypoint. The results are shown in Figure 13.

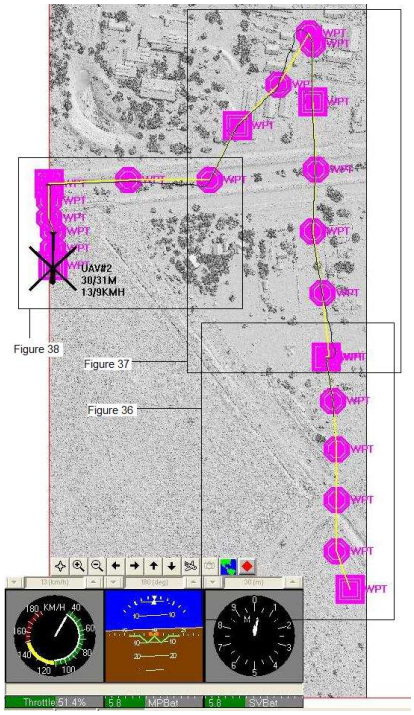

Figure 13 Results with hover to command

D. Test Case 8: Air sampling/data collection with Digital Elevation Map included

One limitation of the previous test cases is that the map loaded and used in the ground station has no elevation data. This test case explores the results of the simulation when a conversion and insertion of the LiDAR map into a Digital Elevation Map suitable for the ground station is included. A software was written to convert the LiDAR LAS files into a suitable format for the ground station software. That was done by decreasing the LiDAR resolution using LAS tools and then converting from LAS format to USGS DEM format through GIS GRASS. Figure 14 shows the DEM imported and inserted inside the UAV ground station software. Figure 15 shows that the simulation (and the UAV) crashes (as expected) when the UAV is asked to fly over a location with an elevation that is below a given threshold. In figure 16, it also possible to observe that there is a small offset between the geotif map (loaded on the Google earth image of the same zone) and the DEM. The offset is due to the resolution selected in the coordinate convertion of the map. A solution to this was to rotate and align the maps and waypoints from the optimal path.

From the analysis of the simulations with a digital elevation model, it is then possible to define no-fly zones that cover the areas more dangerous for the flight (in terms of number and high of obstacles) or prohibited by the regulation about UAV. Figure 16 shows an example of a no-fly zone over one of the buildings mapped by the LiDAR LAS files.

Figure 14. –Digital Elevation Map (DEM) inside the UAV ground station software.

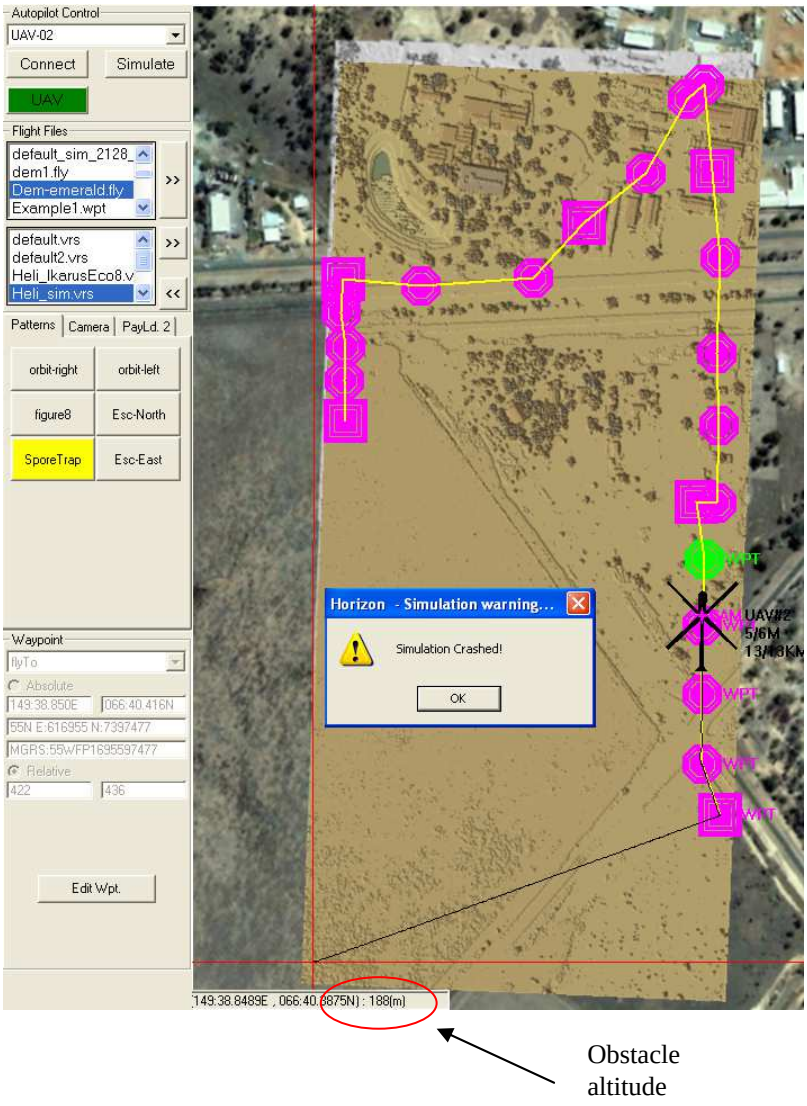

Figure 15. Optimal path plan and simulation with Elevation Map (DEM).

Figure 16. Example of no-fly zone within the map

VI. CONCLUSIONS

This research proposes a method of obtaining an optimal path with LiDAR-augmented information and output to a UAV autopilot. The shortest path from a start to an end point can be successfully found and plotted using LiDAR data and the A* algorithm. The total time taken for the computation is approximately 5 seconds (without the MATLAB or adding the optimal path to the LiDAR map visualisations). A path can also be found when nodes are added in between the start and target positions. It is shown through flight simulation that the path found can successfully be followed by a helicopter UAV while avoiding obstacles and performing tasks such as sampling air and acquiring data transmitted from the ground.

There are a few limitations of this work; one is the use of a 2D algorithm which finds an optimal path for which there may be a height restriction, ongoing work focuses on exploring complex 3D algorithms. Another issue is the capabilities of the ground computer and onboard computer to process the LiDAR data, as the computer capabilities are increased larger maps can be used.

ACKNOWLEDGEMENTS

The authors would like to thank to the Open Source Geospatial Foundation, Martin Isenburg and Richard Horne, for the developing of open source initiatives such as GRASS GIS, LAStools and 3DEM respectively. This research was supported by the Queensland University of Technology (QUT) vacation research scholarship program. We would also like to acknowledge the GRASS, LAStools and 3DEM sued in this research.

REFERENCES

1. Jones, D.I., Golightly, I.T., Roberts, J., Usher, K., and Earp, G.K., Power Line Inspection - A UAV Concept. in The IEE Forum on Autonomous Systems. 2005. London, United Kingdom: IEE.
2. Kelcey, J.; Lucieer, A. Sensor correction of a 6-band multispectral imaging sensor for UAV remote sensing. Remote Sens **2012**, 4, 1462–1493.
3. Berni, J.A.J., Zarco-Tejada, P.J., Suárez, L., and Fereres, E., Thermal and narrow-band multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. IEEE Transactions on Geoscience and Remote Sensing, 2008. 3(47): p. 722–738.
4. Zhou, G., Ambrosia, V., Gasiewski, A,J., and Bland, G., IEEE, Special Issue on Unmanned Airborne Vehicle Sensing Systems for Earth Observations. IEEE Transactions on Geoscience and Remote Sensing for Earth Observations.
5. Lee, D.S., Gonzalez, L.F., Srinivas, K., and Periaux, J., Robust evolutionary algorithms for UAV/UCAV aerodynamic and RCS design optimisation. Computers & Fluids, 2008. 37(5): p. 547-564.
6. Gonzalez, L.F., Whitney, E.J., Periaux, J., Sefrioui, M., and Srinivas, K. (2004). A robust evolutionary technique for inverse aerodynamic design. Design and Control of Aerospace Systems Using Tools from Nature. In Proceedings of the 4th European Congress on Computational Methods in Applied Sciences andEngineering. Vol. 2, pp 24-28
7. Kok,J. Gonzalez, L.F, Kelson, K. FPGA implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning Evolutionary Computation, IEEE Transactions on 17 (2), 272-281.
8. Periaux, J. Gonzalez, L.F, Whitney,J.W., Srinivas, K. MOO methods for multidisciplinary design using parallel evolutionary algorithms, game theory and hierarchical topology: Practical application to the design and optimisation of

UAV systems (Part 1). Von Karman Institute (VKI) Lecture Series, Introduction to Multidisciplinary to Optimisation and Multidisciplinary Design: Applications to Aeronautics and Turbomachinery, March 6–10, 2006.

9. Lee, DS, Gonzalez, L.F., Whitney E.J. Multidisciplinary Multi-fidelity Design tool: HAPMOEA-User Guide, 2007
10. Gonzalez, L.F., Whitney, E.J., Srinivas, K., and Periaux, J., "Optimum Multidisciplinary and Multi-Objective Wing Design in CFD Using Evolutionary Techniques," Proceedings of International Conference on Co mputational Fluid Dynamics 3, Toronto, Canada, July 12-14, 2004
11. Borenstein, J. and Y. Koren, Histographic in-motion mapping for mobile robot obstacle avoidance. IEEE Transactions on Robotics and Automation, 1991. 7(4): p. 535–539.
12. Bortoff, S.A. Path planning for UAVs. in American Control Conference, 2000. Proceedings of the 2000. 2000.
13. Jain, R. and A. Koronios, Innovation in the cluster validating techniques. Fuzzy Optimization and Decision Making, 2008. 7(3): p. 257-267.
14. LaValle, S.M., Planning Algorithms, ed. C.U. Press. 2006, Cambridge, U.K.
15. Sathyaraj, B., Jain, L., Finn, A., and Drake, S., Multiple UAVs path planning algorithms: a comparative study. Fuzzy Optimization and Decision Making, 2008. 7(3): p. 257-267.
16. Priestnall, G., J. Jaafar, and A. Duncan, Extracting urban features from LiDAR digital surface models. Computers, Environment and Urban Systems, 2000. 24(2): p. 65-78.
17. Standards for digital elevation models, U.S. Department of the Interior, U.S. Geological Survey, National Mapping Division.
18. Rodriguez, E., Morris C.S., Belz J.E., Chapin E.C., Martin, J.M., Daffer, W., and Hensley S., An assessment of the SRTM topographic products Technical Report. 2005, Jet Propulsion Laboratory Pasadena, California. p. 143.
19. Drury, S.A., A Guide to Remote Sensing: Interpreting Images of the Earth, ed. O.S. Publications. 1990.
20. Wehr, A., Topographic Ranging and Scanning: Principles and Processing. , ed. B.R.C. Press. Vol. ch. 4. 2009. 129-131.
21. Isenburg, M., Liu, Y., Shewchuk, J.R., Snoeyink, J., and Thirion, T., Generating Raster DEM from Mass Points Via TIN Streaming in Geographic, Information Science, S.B. Heidelberg, Editor. 2006. p. 186-198.
22. Liu, X. and Z. Zhang. LIDAR DATA REDUCTION FOR EFFICIENT AND HIGH QUALITY DEM GENERATION. in XXI Congress of the International Society of Photogrammetry and Remote Sensing (ISPRS2008). 2008. Beijing, China.
23. Miniature Aircraft 'X-Cell 60' kit review. . 1995 [cited January 20, 2010]; Available from: http://www.iroquois.free-online.co.uk/xl60.htm
24. Micropilot, HORIZONmp User Guide.